WPF大基本功

Application的生命周期

OnStartup: 表示启动应用程序时

OnActivated: 表示激活应用程序时

OnDeactivated: 表示由激活状态变为非激活状态时

OnExit: 表示退出应用程序时

Window窗体的生命周期

SourceInitialized创建窗体源时引发此事件

Activated当前窗体成为前台窗体时引发此事件

Loaded当前窗体内部所有元素完成布局和呈现时引发此事件

ContentRendered当前窗体的内容呈现之后引发此事件

------Deactivated当前窗体成为后台窗体时引发此事件 ====Activated当前窗体成为前台窗体时引发此事件

Closing当前窗体关闭之前引发此事件

Deactivated当前窗体成为后台窗体时引发此事件

Closed当前窗体关闭之后引发此事件

Unloaded当前窗体从元素树中删除时引发此事件

ContentControl

Content属性在 ContentControl类中

加载和解析 XAML

编译器会将 XAML 文件（如 MainWindow.xaml）编译为 BAML (Binary Application Markup Language)，这是一种压缩和优化的二进制格式。

运行时，CLR 会加载 BAML 并将其解析回一个对象树。这个过程会：

实例化 XAML 中定义的每一个元素（如 <Window>，<Grid>，<Button>）为 .NET 对象。

设置每个对象的属性（如 Width, Height, Content）。

根据元素的嵌套关系，建立起逻辑树 (Logical Tree)。

对象初始化过程中，会按顺序引发一系列事件

Initialized: 元素已被初始化并从 XAML 中加载了基本属性。此时逻辑树已建立，但尚未进行布局测量和排列。

Loaded:

整个窗口（或页面）已加载完毕，逻辑树完整，并且即将被呈现到屏幕上。这是通常进行最终初始化操作（如加载数据）的地方。

- 逻辑树 **(Logical Tree)**：反映 XAML 中声明的控件层次结构，用于核心功能如属性继承、资源查找、事件路由。
- 可视树 **(Visual Tree)**：描述控件视觉结构的更底层、更详细的树，用于渲染和命中测试。你可以使用 `VisualTreeHelper` 来遍历它。
- 布局系统 **(Layout System)**：递归的 测量 **(Measure) ->** 排列 **(Arrange)** 过程。任何影响布局的变化（如窗口大小调整、内容更改）都会触发一个新的布局传递 (Layout Pass)。
- 保留模式图形 **(Retained Mode Graphics)**：与“立即模式”（如 WinForms 的 `Paint` 事件）相反，WPF 维护一个可视化对象列表（可视树），并由系统负责在需要时重绘它们。开发者只需声明“要什么”，而不需关心“何时画”和“怎么画”。

控件基类路线

Button->ButtonBase->ContentControl->Control->FrameworkElement->UIElement->Visual->DependencyObject->DispatcherObject

StackPanel->Panel->FrameworkElement->UIElement->Visual->DependencyObject->DispatcherObject

Rectangle->Shape->FrameworkElement->UIElement->Visual->DependencyObject->DispatcherObject

WPF用启动时，分别用于呈现界面（后台线程）和管理界面（UI线程）。

DispatcherObject的 Dispatcher调度员提供了 Invoke和 BeginInvoke两个方法，供我们可以安全的访问 UI线程中的控件

版本号 #3
由 叶子 创建于 19 三月 2026 09:43:13
由 叶子 更新于 19 三月 2026 10:17:59